Bolt Beranek and Newman Inc.

(12)

BBN Report No. 3451

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Quarterly Progress Report No. 8
1 August 1976 to 31 October 1976

March 1977

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2901, Contract No. N00014-75-C-0773.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 3451
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES
5. TYPE OF REPORT & PERIOD COVERED: 8/1/76 - 10/31/76
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): R. Schantz, R. Thomas
8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0773, ARPA Order 2935
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc., 50 Moulton Street, Cambridge, Massachusetts 02138
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: BBN-3451
12. REPORT DATE: March 1977
13. NUMBER OF PAGES: 20
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distributed computation
National Software Works
distributed operating system
TENEX operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes ~~BBN~~ efforts in the design of the National Software Works system and ~~BBN~~ efforts to integrate TENEX into the National Software Works system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

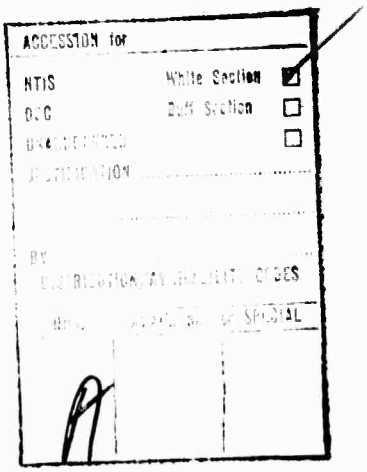
Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100

TABLE OF CONTENTS

		Page
1.	Introduction.	1
2.	MSG: The NSW Interprocess Communication Facility . . .	3
	2.1 Process-to-MSG Interface	3
	2.2 MSG Reliability Improvements	6
	2.3 MSG Operational Aids	7
3.	The Foreman: Supporting NSW Tool Execution	11
	3.1 Augmented Interaction With Other Components. . . .	11
	3.2 Functional Additions and Upgrades.	13
	3.3 Encapsulation and Additional NSW Tools	16
4.	TENEX FE Dispatcher Modifications	19



1. Introduction

This quarter we continued our participation in the National Software Works (NSW) system development effort. The major thrust of our activities in this area remains focussed on implementation efforts to achieve a functional, usable NSW. Along with the continued development of several major system components, we also participated in a number of contractor/sponsor meetings and briefings, which are so necessary in coordinating this type of multi-contractor program. This quarter there were meetings with Tool Bearing Host implementors from MIT and UCLA to discuss and clarify our previously issued component specification documents, component integration sessions with COMPASS personnel, and discussions with GSG personnel regarding the demonstration of the NSW technology.

Our continuing implementation effort centered around the TENEX MSG component (the NSW inter-host communication facility, see BBN Report No. 3237 and previous QPRs), the TENEX Foreman component (the NSW component for running tools, see BBN Report No. 3266 and previous QPRs), and the TENEX Front End Dispatcher. The following sections detail the major implementation enhancements for these modules.

In another on-going effort, we are continuing discussions with Digital Equipment Corporation (DEC) and ARPA toward

achieving a convergence between the current TENEX operating system and future releases of the DEC TOPS-20 operating system. This effort is intended to enhance the possibility of directly utilizing TENEX NSW software in making a DEC TOPS-20 ARPANET host function as an NSW Tool Bearing Host (see the previous QPR, BBN Report No. 3450). Proposals for changes to the DEC TOPS-20 to accommodate NSW are currently being reviewed and modified to suit the various needs and goals of the concerned parties.

2. MSG: The NSW Interprocess Communication Facility

Work on the implementation of MSG has progressed in three areas. First, some new features were added at the process-to-MSG interface. These features were added in response to newly identified NSW needs. Second, MSG was made less vulnerable to remote host and transient network failures. Third, new features were added to support NSW system debugging, system integration, and system operation. The remainder of this section describes these three areas in more detail.

2.1 Process-to-MSG Interface

A new MSG process primitive that returns the MSG name for the executing process has been implemented. This primitive, called "whoAmI", was not included in the original MSG design specification. It was added, in part, to satisfy a requirement of the NSW File Package processes. In particular, when a Works Manager process instructs a File Package process to move a file to a particular workspace on a particular host, the File Package process must be able to determine whether a network file transfer or a purely local file copy operation is required. Although some host operating systems provide means a process might use to determine its local host identity, we felt that a standard MSG primitive that could be used for this purpose would insure the availability of this capability on all NSW hosts. The WhoAmI primitive returns to the executing process all four components of

its MSG name: generic process class, host identity, host incarnation number, and process instance number (refer to BBN Report No. 3237 for a discussion of these name components).

The MSG direct connection feature has been enhanced to provide a "connection broken" signal to processes that request it. A process executing the OpenConn primitive to establish a direct connection to another process can specify a PSI channel for the signal. If the connection is spontaneously broken after it has been opened, MSG will notify the process via the specified PSI channel.

This feature is needed to support some of the higher level NSW component protocols, such as the protocol used by Front End, works Manager, and Foreman processes to terminate tool sessions. In addition, it will be used by NSW processes for early detection of failures involving direct connections in order to prevent errors from propagating. For example, the Foreman will request the "connection broken" PSI signal for the TELNET connection it establishes to support Front End-to-tool communication. If signalled that the connection has been (unexpectedly) broken, the Foreman will act to "detach" the tool for possible later "re-attachment" from the Front End or eventual termination by the works Manager. This action by the Foreman will prevent the broken connection condition from being propagated to the tool which may be unprepared to handle it.

we have completed initial implementation of a user TELNET package for use with MSG direct connections. Previously, MSG provided minimal support for user TELNET connections. It would act only to establish and break these connections. To use the TELNET connections the process itself (e.g., a Front End process) was required to perform all of the TELNET protocol: interpret TELNET control characters, negotiate TELNET options, etc. (see ARPANET document NIC No. 7104 for specification of the TELNET protocol).

The new TELNET package was constructed primarily to support NSW Front Ends as well as more general user TELNET programs. It provides means for a process to "connect" and "disconnect" a terminal (the NSW user's terminal) to previously established user TELNET connections. When a terminal is "connected", the TELNET package acts to pass characters typed through to the TELNET connection and to print characters received from the TELNET connection. In doing so, it performs the TELNET protocol functions on behalf of the process. The TELNET package can be used by a process to control multiple TELNET connections. This capability allows a Front End to maintain several concurrent tool instances for a user and, at the user's instruction, to dynamically "connect" the user's terminal first to one tool instance and then to another. When the terminal is not directly coupled to a tool instance (through the corresponding TELNET connection), the TELNET package buffers any output that may arrive from the tool. The buffered characters can be printed

when the user redirects his attention to that particular tool instance.

The user TELNET package is implemented in BCPL and MACRO-10. It is based on the connection control routines used in the standard TENEX user TELNET subsystem. We modularized these routines and implemented interfaces to them in order to permit their use with other programs in a flexible way.

2.2 MSG Reliability Improvements

The MSG modules which support inter-host communication have been made more resilient in their ability to handle failure situations. The overriding consideration in implementing the first version of MSG was to produce a useable communication facility as quickly as possible so that debugging and checkout of the NSW system components could begin. During the initial implementation effort, whenever a choice had to be made between providing increased functionality or increased failure recovery capability, the choice was always to spend our limited resources on increased functionality. Consequently, the TENEX MSG was very vulnerable to certain remote host and network failures. In some situations the failure of one remote host could interfere with the ability of MSG to communicate with other remote hosts. In others, it was possible for an ill-timed remote host failure to interrupt MSG communication within the local host. The problem here was that MSG was very limited in its ability to recover from

spontaneously broken MSG-to-MSG connections (see BBN Reports No. 3237 and 3315 for descriptions of the use of these connections).

At present MSG provides sufficient functionality to support the immediate needs of the NSW system component implementers. During this quarter we have improved its failure recovery capability. MSG has been made much less vulnerable to remote host failures by improving its ability to detect and recover from unexpected broken MSG-to-MSG connections. The following procedure is used whenever an MSG-to-MSG connection breaks while MSG is using it to send a message to a remote MSG: the message in question is requeued for later transmission to the host; the broken connection is discarded; and, an attempt to re-establish communication with the host is made. This procedure insures that any message destined for the host will be delivered if communication with the host is re-established before the "pending event" associated with it is timed out. If a connection breaks as MSG is reading a message from a remote host, the procedure is to discard the partially received message, to discard the broken connection, and to attempt to re-establish communication with the host.

2.3 MSG Operational Aids

Features to facilitate the configuration of NSW systems have been added to MSG. These features make it easy for system implementers and the NSW operations staff to configure an

instance of MSG. By an "instance" of MSG we mean a collection of cooperating, intercommunicating MSG modules on a set of host computers together with the processes these modules support. It is possible to imagine independent collections of MSG modules (i.e., separate MSG instances), each collection of which is dedicated to a separate application. For example, there might be the operational NSW configuration and one or more different NSW configurations being used for checkout and integration of new versions of system components.

An MSG configuration is defined by two declarations:

- a declaration of the generic process names supported by the instance (e.g., Works Manager, Front End);
- a declaration of the hosts the comprise the configuration.

The generic names known to MSG are declared in a text file which is read as part of MSG initialization. Each process class declaration includes the generic class name together with process creation and termination information. The creation information specifies how generically addressed messages for the process class are to be handled. For process classes implemented on remote hosts, it is merely the host that supports the process. For locally supported process classes, the creation information includes the name of the TENEX SAVed file for the process as well as information that specifies exactly how processes in this class are to be created to handle generically addressed messages. This later information specifies details such as whether the process should be in the same job as other processes in the class, the

maximum number of processes in this class that may exist simultaneously, etc. For locally supported processes, the termination specification details what action MSG is to take when the process executes the StopMe primitive: e.g., kill the TENEX fork that implements the process, reuse the TENEX fork for another new process in the same class, log out the job that supported the process if no other processes in the class exist, etc.

The hosts that comprise an MSG configuration are declared in a second text file. For each host in the configuration the MSG ICP contact socket is specified. This socket is the contact socket used to establish MSG-to-MSG connections (See BBN Report 3237). By reserving a separate ICP contact socket for each MSG instance on a host, it is possible for several distinct MSG modules, each dedicated to a particular MSG configuration, to co-exist on the same set of hosts without interfering with one another.

Last quarter we reported on an MSG process monitoring and debugging capability. This capability is supported by a command language interpreter that is invoked when a user (i.e., an implementer of an NSW component such as the Works Manager or Foreman) types an appropriate control character (CNTL-S). A number of new commands have been added. These include commands to allow users to start and stop processes as well as to invoke debuggers for processes. For example, when a user starts a

process he has the option of specifying a debugger (DDT, BDDT, IDDT) or not. If no debugger is specified, the process is started normally. If one is specified, the debugger for the process is started and the user is free to set break points, etc. before placing the process into execution. Should the process subsequently terminate abnormally (e.g., because it executed an illegal instruction), MSG will automatically activate the debugger for it if one has been specified. Otherwise, MSG will simply report the termination. In this case, the user can explicitly invoke a debugger for the process via an MSG command if he wishes to examine the process.

Other commands have been added this quarter. There is a command which prints MSG configuration information. This includes the generic names known to MSG, process creation specification information, and the status of hosts in the configuration. In addition, there is a command to invoke a debugger for MSG itself. This is useful to us as MSG implementers and maintainers.

3. The Foreman: Supporting NSW Tool Execution

During this reporting period, the TENEX NSW Foreman component was substantially upgraded to permit it to be used operationally in an on-line experimental NSW configuration. As described in our previous report (BBN Report No. 3450), the initial version of the TENEX Foreman for a multi-host NSW configuration was a minimal facility. It was used mostly to "shakedown" the coordination of interactions between NSW components. The changes and additions incorporated this quarter augmented the Foreman's adherence to the specification of the Foreman design document, and automated many of the resource allocation and error detection decisions which need to be handled in an operational environment. The major Foreman changes instituted this quarter, and a discussion of their importance comprise the remainder of this section.

3.1 Augmented Interaction with Other Components.

The TENEX MSG component was recently upgraded to include the support for direct ARPANET connections between NSW components. Generally, when a tool is run, it requires a direct connection to the NSW Front End component. This connection (usually a Telnet connection) was previously established via ad hoc conventions between the FE and Foreman. As soon as the direct connection facility was operational within MSG, the Foreman and FE were changed to make use of MSG for all of their communication needs. In a related area, the TENEX MSG facility added a component

self-identification primitive, the WhoAmI function. This function returns to the invoking process its complete MSG process name including the component type (e.g., Foreman), its local host, the incarnation number for MSG on this host, and the instance number of the component. The Foreman now uses the MSG process name as a unique identifier for the various instances which may be running. Unique names are needed in circumstances such as identifying the process which currently owns some resource. Additionally, the MSG incarnation number is also used in the Foreman program code to help ensure proper initialization of data bases and process synchronization after a system restart.

Another notable achievement this quarter has been the redesign and reimplementation of the protocols for handling NSW tool initiation and termination. The original tool initiation and termination scenarios were very synchronous and often roundabout in routing information. They reflected a different design philosophy than is currently used. Thus, those scenarios were extremely far from optimum in areas such as elapsed time to completion, number of messages sent, etc. The redesigned scenarios take advantage of parallel execution where possible (especially in sending MSG messages), and in addition try to minimize the number of messages sent in achieving orderly tool initiation and termination. By having the Foreman component simultaneously try to open a connection to the FE while notifying the WM of the parameters selected for running the tool, we help minimize the delay seen by the user after requesting a tool to be

executed. Similarly, by attempting to orderly remove any network connections, while at the same time reporting tool use changes to the WM, we try to minimize the delay involved in tool termination.

These new scenarios were implemented for the TENEX Foreman (as well as for the TENEX FE and WM by Massachusetts Computer Associates) and integrated with similarly updated components. These scenarios are the ones currently in use in the experimental system running at ISI, and will be completely documented in the next revision of the Foreman specification document.

A further operational change incorporated this quarter was the use of message timeouts to protect against indefinite delays due to component outages and malfunctions. Currently, the timeout merely signals the Foreman to do an orderly shut down and remove itself from the NSW operation. Ultimately, recovery procedures will be incorporated to ensure no loss of NSW files due to component outages. The message timeouts can optionally be made to be effectively infinite so as to facilitate the use of the Foreman in a debugging environment.

3.2 Functional Additions and Upgrades.

During this quarter, we designed and implemented the Foreman functions for accessing and manipulating workspaces for running TENEX NSW tools. Previous to this, and as an interim measure, all tools ran out of a single TENEX directory and could therefore

interfere with other running tools. Our multi-workspace design is based on the static creation of a number of TENEX host directories and login names which are to be used to run tools. A shared data base (in the form of a TENEX shared file) contains information pertaining to the workspace directories and their current availability. Basically, when a Foreman is started (as a result of receiving a Generic request for tool service), it accesses the data base (exclusively) and finds a free workspace. It then marks the workspace entry in the data base as "taken" (and by whom), and releases use of the data base. To run the tool in that workspace, the Foreman first connects to that directory and simulates a login so as to allow the TENEX operating system to take responsibility for preventing tool access to other TENEX file spaces. Before the tool is run, the workspace is cleared of all residue files from any previous use of the workspace, to assure data privacy and avoid possible name or resource conflict.

Since the workspace descriptor data base exists on a non-volatile storage medium (i.e., disk memory), after a system restart (or crash), its contents are usually the same as before the crash. That is, workspaces which were marked as being in use will still be marked that way. This is very important in the long-term reliability plans, but presents a problem in the short term. The initial releases of the NSW will not attempt to reclaim user files which are lost in a tool workspace due to some malfunction. However, since there are only a limited number of

workspaces, the Foreman must reclaim those workspaces which were in use when the system went down. Failure to reclaim these spaces would result in reduced TBH support for running tools, ultimately preventing any tool from being executed.

As noted in our previous report, we have introduced into TENEX MSG a facility for alerting an MSG process to a system restart. The TENEX Foreman uses this signal to perform the necessary workspace cleanup. The current implementation merely clears any workspace locks that were set when the system stopped. This allows each workspace to be used for running tools in the restarted NSW. More extensive recovery procedures are expected to be designed and implemented for future NSW system releases, and may include resuming a tool session on system restart.

In conjunction with the use of the workspace definition file by the Foreman, we also designed and implemented an initial version of a stand-alone (i.e., non-NSW) program to create and initialize new workspace definition files, and to modify existing files with the addition or removal of potential workspaces. This program takes explicit account of the need to be able to run multiple NSW configurations, even on a single host computer. Simultaneous component debugging and individualized computer resource procurement are but two of the reasons for introducing the concept of multiple instances of an NSW system. The workspace definition file creation program adapts to multiple NSW configurations by permitting the partitioning of available

workspace directories. Currently, a partition is statically assigned to an NSW configuration. That is, when creating or updating the definition file, the user assigns workspaces to the various systems. At some time in the future, we hope to be able to handle dynamic sharing of workspace directories, assigning them as they are needed by the various systems. An important consideration in any such dynamic sharing is the guarantee of a minimum level of service to each configuration. Without such a guarantee, the ultimate utility of dynamic workspace sharing is greatly diminished.

3.3 Encapsulation and Additional NSW Tools.

The TENEX NSW tool encapsulator is that part of the Foreman responsible for overseeing the NSW operation of programs which normally run in the TENEX operating system environment (see BBN Report No. 3266 for further details). There are already a substantial number of TENEX tools which are capable of running in the NSW under encapsulation. This quarter we have validated the use of four more tools for use within NSW. The newest additions are MRUNOFF (Multics type Runoff), a versatile text formatting program, NETSTAT and HOSTAT, programs which dynamically report on the status of the ARPANET and its constituent host computers, and EMLOAD, a TENEX based program loader for creating UYK-20 computer system compatible load modules. The EMLOAD tool is an important addition in providing a complete, consistent tool set for NSW testing by UYK-20 programmers at the Naval Electronic Laboratory Center.

The encapsulator itself has been upgraded in a number of areas, partially to support the existing tools. The domain of encapsulated systems calls has been extended to include some of those functions interrogating the program runtime environment, e.g., the connected directory for the tool, the terminal number for running the tool, etc. While these operations are not used by many tools, the interpretation of these aspects of a computation are changed under NSW. Accordingly, encapsulation had to be extended to cover the additional areas within the context of a TENEX computation. This change typifies our experience with encapsulation. As more tools are introduced, they invariably use TENEX features which were not important to previously installed tools. Thus, the installation of new tools frequently requires modifications in the encapsulation module to more accurately map the TENEX operating environment onto the NSW operating environment.

Other additions made to the TENEX encapsulator module this quarter include the handling of a "user terminal designator" within NSW, and the augmentation of the code to handle the use of non-NSW files by NSW tools. As a result of the first change, a TENEX NSW tool (and hence an NSW user) can ask for a reference handle to "the user's terminal" (designated in NSW as +TTY). In the current implementation requests referring to the "user's terminal" are routed to the ARPANET connection which links the tool to the user's Front End. If not handled in this special fashion, the operation would fail, since there is no "user

terminal" for the (detached) Foreman job. The second change allows partial file names to be specified in naming external files which can be legally accessed by an NSW tool. If a tool requested file is within the class of files named via one of the partial path names, the access is allowed. If it is not within one of those classes, the request is rejected. Previous to this change, each referenced external file had to be completely named at the time of tool installation, and the tool had to be re-installed if a file name was slightly modified. This Foreman change should greatly reduce the effort involved in tool installation and provide a more flexible runtime environment for encapsulated tools.

4. TENEX FE Dispatcher Modifications

The program which automatically dispatches user requests to access the NSW to an appropriate TENEX job running an NSW Front End, was also substantially upgraded this quarter. Our initial experiences with an operational NSW lead to a number of improvements in both the interface to the dispatcher facility and in handling frequently occurring, but out of the ordinary situations.

The dispatcher interface to the NSW user was modified to include more information in its herald, and adjustments were made to coordinate the initial terminal mode settings with the actual use of these network connections. Additionally, the dispatcher was modified to automatically detach itself on command from the NSW operator. Since the dispatcher's normal state is as a detached job, this simplifies the task of bringing up an NSW configuration.

A major effort was expended in the area of dispatcher resource allocation and deallocation, especially when deallocating network virtual terminals (NVT) from NSW use. The initial use of NSW in a test bed configuration revealed that the connection from the user's computer to the dispatched FE job was frequently breaking. This was only in part due to the vagaries of the network. More often than not, a user would simply close his connection to the FE whenever the NSW components acted too slowly or malfunctioned. As a consequence of this, and because

of an apparent bug in the TENEX controlling terminal code, these NVTs were often not deallocated on abnormal termination. This resulted in the system either running out of available NVTs, or else working with a diminished supply. Once the situation was diagnosed, corrective action was taken in both the dispatcher and in the TENEX MSG to assure that the dispatched FE job becomes aware of the disconnected terminal, that the terminal connection is appropriately deallocated, and that the job disappears after allowing the effected components to clean up. This effort was a major step forward in achieving a robust NSW, which can adequately deal with abnormal situations without the need for human intervention.